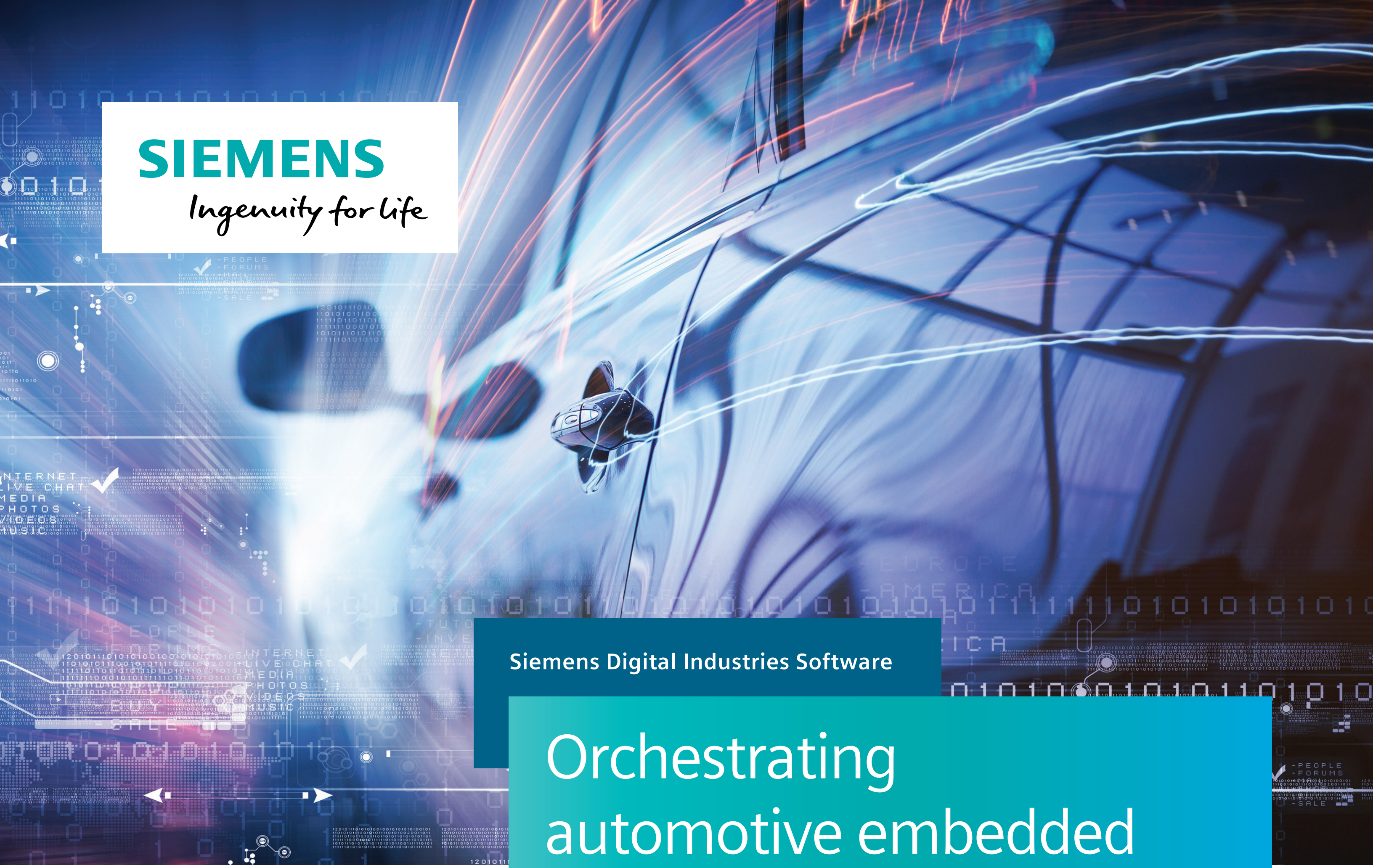

SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Orchestrating automotive embedded application development

Application definition and planning

Executive summary

The automotive industry is undergoing rapid change, driven by an ever-increasing demand for advanced features that rely on sophisticated electronics and embedded software, such as advanced driver assistance systems (ADAS) and connected car features. Embedded application development is a maze-like endeavour where many different tools and teams must try to collaborate under constantly shifting constraints. A unified software application engineering platform can ensure that constraints and requirements are communicated completely and effectively, and that development work satisfies requirements accurately.

Piyush Karkare

Introduction

The automotive industry is a highly dynamic and fast-changing environment. An ever-increasing demand for advanced features that rely on sophisticated electronics and embedded software, such as advanced driver assistance systems (ADAS) and connected car features, are driving forces behind this change. Such is the demand that the core automotive competencies are shifting from mechanical systems to software and electronics. Differentiation and innovation in the highly competitive automotive market increasingly relies on software and electronics, rather than mechanical systems.

OEMs and suppliers are responding to the digital transformation of modern vehicles by standardizing software features across their vehicle lineup. Volkswagen recently announced plans for a new software group, eventually consisting of five-thousand software experts that will develop a uniform operating system across their brands (Automotive News Europe, 2019).

Amid this evolution, new players from the technology sector, such as Waymo and Apple, are entering the automotive and mobility market. These companies are well versed in designing and building powerful software features, applications, and user experiences. At the same time, automotive and mobility startups are launching in the hundreds and looking to merge tech and automotive experience from the start to create compelling vehicle platforms. In both cases, the growing importance of vehicle software and electronics has leveled the playing field with legacy automotive brands.

In addition to these new challenges, competition between established automotive brands is more intense than ever. In 2001, the Ford Escape had around six direct rivals in the small crossover SUV market. Today, there are more than twenty-one competitors in this segment form mainstream and high-end luxury manufacturers alike. Competition drives the need for greater vehicle content (in the form of new features), and greater content drives complexity.

This complexity manifests overwhelmingly in the electrical and electronics (E/E), and software domains. This can be seen in the number of requirements and specifications in these domains, which is significantly higher than others (figure 1). For example, a well-equipped mid-range vehicle will have about 50,000 mechanical and regulatory requirements. The E/E and software domains on that same vehicle will contribute upwards of 450,000 requirements. The E/E and software verification and test plans multiply the number of items that need to be tracked several times over.

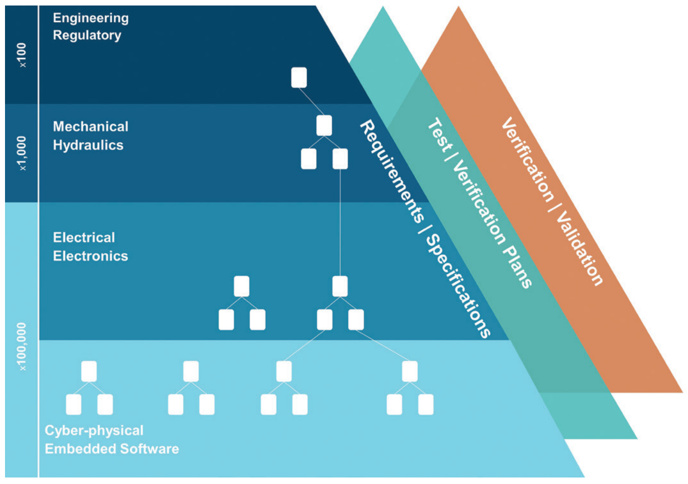

Figure 1: The requirements and specifications for the software and E/E domains are significantly more complex and numerous than other domains.

While E/E and software intra-domain complexity is significant, cascading and enforcing requirements across domains is potentially even more complicated. The E/E, software, and mechanical domains are highly inter-dependent with many points of interaction. Requirements often have cross-domain relationships that can be extremely difficult to understand, let alone predict. However, it is important that teams understand each design requirement and how it affects the larger system.

While OEMs and suppliers attempt to build processes and value-chains, design changes and product evolution are constant. What matters is the ability to maintain consistency and quality among related data across the domains. To do this, the key is ensuring overall consistency and compatibility between the work that various stakeholders are performing across many platforms and organizations. The OEM and its suppliers must be able to coordinate their activities through each level of design abstraction, from the vehicle platform-level features, to the logical allocation and actual software application development.

OEMs are moving towards service-oriented architectures. This transition is characterized by consolidating ECU hardware into a smaller number of domain control units, leaving services and functionality to be accomplished through software ported on ECU abstractions. Such architectures make it extremely prudent to track where each service is running to enable proper application-level decompositions. A feature-function architecture with decompositions into services and mappings into ECU abstractions is an effective, organic, and scalable approach for tracking changes and their effects.

Coordinating activities between any of these teams or abstractions can be a significant challenge, but organizing embedded application development is particularly difficult. Automotive embedded application development is a maze-like endeavour where many different tools and teams must try to collaborate under constantly shifting constraints. This environment makes collaboration a challenge, leading to problems. Today, software systems are a major source of program risk due to the complexity and criticality of software applications. Software defects and frequent changes in constraints cause production delays, budget overruns, and increased warrantee costs.

For the purposes of this paper, automotive embedded software application development can be divided into three segments:

- Embedded application definition and planning
- Embedded application development and quality assurance
- Embedded application delivery and monitoring

In this paper, we will focus on the first segment–embedded application definition and planning.

Application definition and planning

Some OEMs have successfully established a feature-centric, architecture driven approach for in-vehicle software. With such an approach, application definition and planning initiate when the vehicle-level feature requirements, specifications, and constraints are fully defined and cascaded to the software application abstraction. These requirements, specifications, and constraints define the vehicle platform or vehicle-level feature design intent (figure 2). For example, the vehicle engineers define specifications and constraints for the vehicle features or functions that enable ADAS features: lane-keep assist, adaptive cruise control, forward collision warning and mitigation, and more.

Downstream, the software engineers start to decompose the feature level definition down to the application-level needs and definitions. These drive the next stage of decomposition, down to specific software functions. In reality, the decomposition of software functions and their standardization across vehicle platforms is a continuous process throughout development (figure 3). Software engineers are constantly

^{SW}F – Customer/engineering feature	^{L}C – Logical component
^{SW}f – Decomposed software function	^{P}C – Physical component
^{HW}f – Decomposed hardware function	

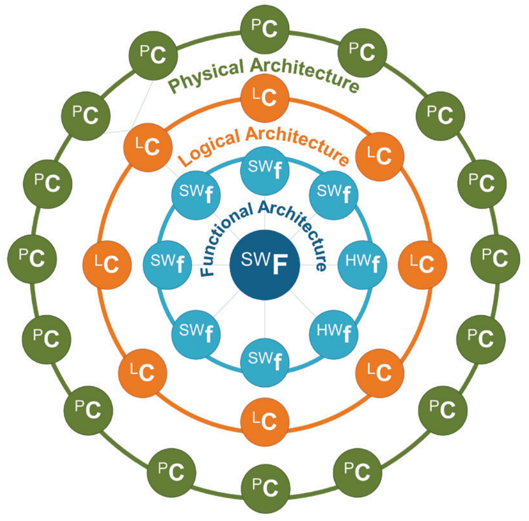

Figure 2: Software features are decomposed into software functions, allocated to logical components, and mapped to physical components with part-numbers.

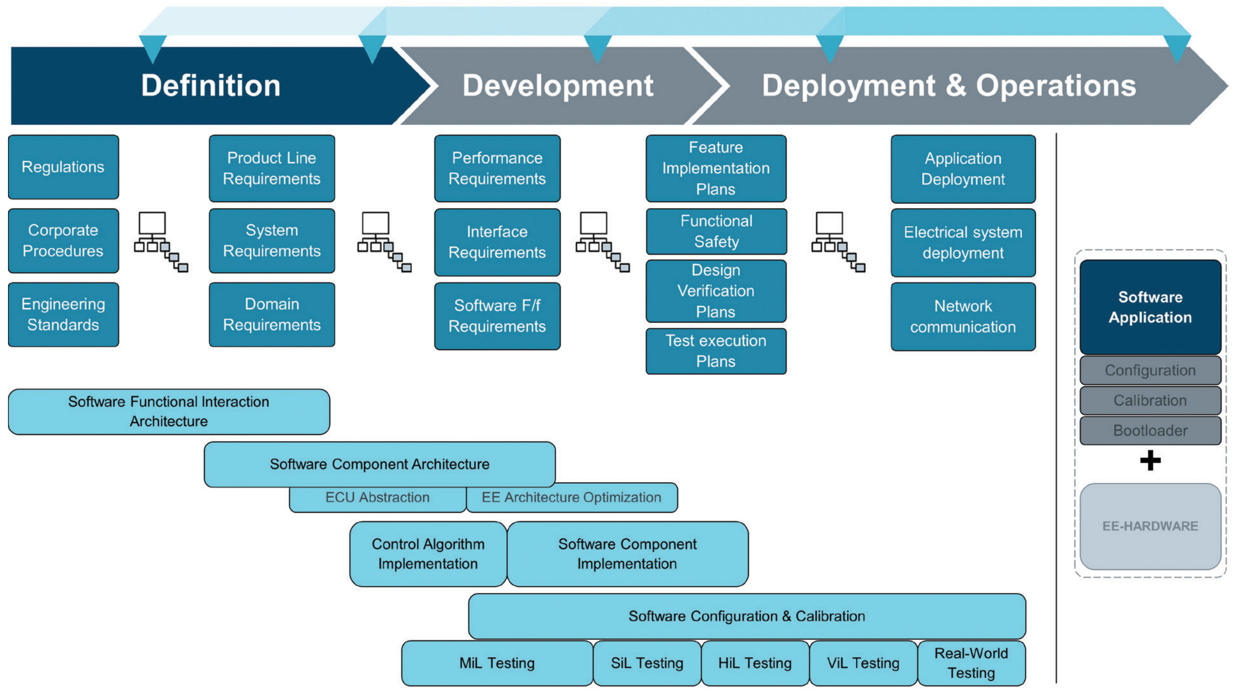

Figure 3: Functional decomposition from requirements and constraints is a continuous process throughout development.

decomposing functions from higher-level requirements to realize a vehicle's feature set. As the engineers develop these functions, they will maximize functional uniformity and optimize functional interactions to ensure the functions can be scaled across vehicle platforms. Where possible, the approach allows engineers to reuse functions from other systems or organizations.

At this stage, it is also critical to consider safety concepts for the application and develop strategies to manage risk. Robust safety requirements and verification and validation strategies are defined at the system level, and decomposed all the way down to the embedded software application development level. Software engineers develop failure modes, effects, and diagnostics analysis (FMEA/FMEDA), hazard analysis and risk assessment (HARA), and other safety requirements and analyses. The engineers document and track these many processes to meet ISO 26262 guidelines.

Key challenges

The complexity involved in embedded application development makes managing the process manually an extremely difficult and time-consuming ordeal (figure 4). Meanwhile, automotive embedded software engineers are under pressure to deliver high quality software that supports the full range of vehicle variants, all on tight timelines. As engineers work to meet this expectation, system-level changes regarding interfaces, functional behaviors, hardware consolidations, hardware limitations, and network bandwidth issues can quickly derail fast-paced software development activities. Tracking such system-level changes is a daunting challenge that can result in making or missing a deadline. Worse, failure to track these changes can cause costly re-work depending on when the changes are prescribed, further delaying development and potentially adding execution risks.

The implementation of system-level changes and updates is a multi-domain and multi-organizational activity. A new specification from the chassis group, potentially caused by an issue with the electrical wiring harness or curb-weight considerations, might impose new requirements on powertrain functions that, in turn, can affect an application under the body-controls group. Tracking and understanding the origination of changes and their implications throughout domains and organizations becomes nearly unmanageable as teams progress through the development lifecycle. As a result, changes may be made to software applications with or without a complete understanding of the change.

ECUs, sensors, and other hardware needed to implement vehicle features may be consolidated, or modified to support different implementations. As variants are developed or as hardware is consolidated, the embedded software application needs to optimize its operation across these hardware variants. To do so, engineers have to capture specifications and requirements from each hardware or operating system to configure the application for optimal functionality across the range of supported hardware variants and operating systems.

The clarity and quality of design requirements, at all levels, often deteriorates as employees retire or move to other companies, and are thus unavailable to clarify missing or ambiguous information. Losing such internal knowledge can result in significant barriers during development. The integrity of software and other models can also suffer as various stakeholders may be working from incorrect or out-of-date models. In fact, some groups may be working independently of the models altogether, distorting the source of truth for a given component in the vehicle. These seemingly small inconsistencies create significant delays and costly rework downstream.

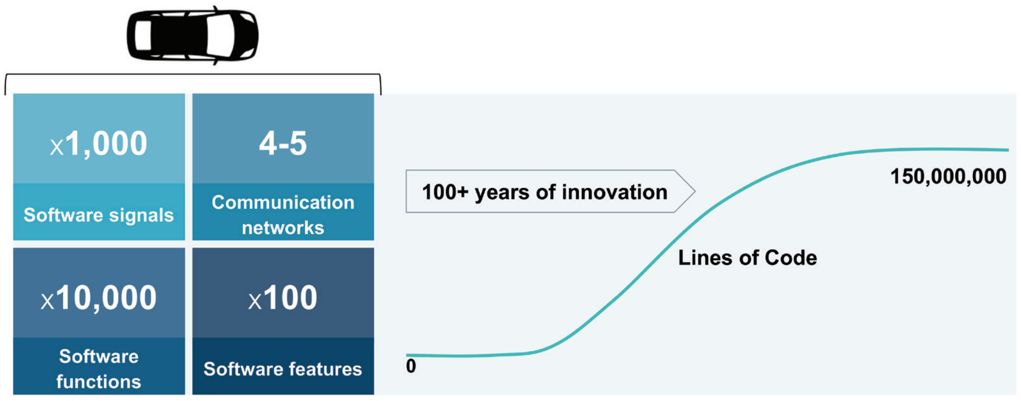

Figure 4: Growing software complexity makes managing application development extremely difficult.

Embedded application definition and planning with a unified platform

Embedded application definition and planning projects need to inhale and track the system-level features or product information, with links between the application and system-level. These links enable active and organic alignment with system-level changes, and ensure the project remains in context of the overall system during development. The definition or change of software requirements triggers software architecture and model changes to align with the new requirements, as well as updates to align with control algorithms. These modeling activities can now be verified and validated at the application and system level for the desired outcome before any code level changes are triggered.

Product integration

A clear line-of-sight from the implementation all the way up to the system-level context is a necessity in complex automotive embedded software domain (figure 5). Software engineers should know how their elements of the application definition tie back to the system-level features definitions and hardware constraints.

Maintaining a system-level feature context is important because it ensures that all project deliverables are consistent with the system-level requirements, constraints, and dependencies. This is true especially as changes are made at the system level. Perhaps a new requirement is defined that combines automatic emergency braking and adaptive cruise control features, targeting implementation on the adaptive cruise control or chassis hardware module. By maintaining a system-level context, software engineers will understand how this change affects the application they are developing, and any changes they need to make. It is imperative that engineers can see these relationships as they work to deliver high quality software that supports multiple vehicle variants at a rapid pace.

A unified, integrated, and extensible platform for embedded application development is critical to help manage the numerous and intricate relationships between software development activities and the system-level definition, across domains and organizations. Such a platform can coordinate requirements to work items, and help manage the cascading and implementation of changes throughout the software architecture.

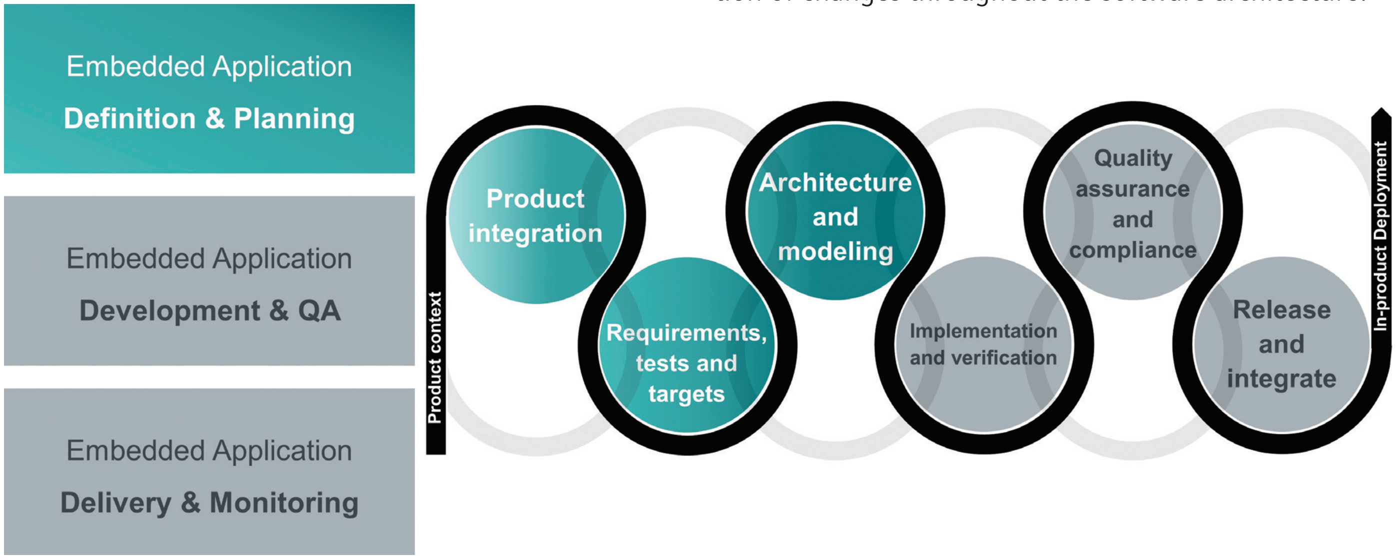

Figure 5: A clear line of sight to the vehicle-level feature definition and hardware constraints is needed throughout the embedded application development process.

Advanced software development coordination tools, such as Polarion, can consume and track the system-level feature definition to create a direct link between system level changes and application development, ensuring the application and the overall system development stay in sync (figure 6). For example, the system definition may call for an application for the power steering control module (PSCM) to manage all the steering control features and functions, such as pull-drift compensation or lane-keep assist, among others. The coordination tool can import the relevant requirements, specifications, vehicle-level models, configuration and calibration parameter definitions, and more from the system-level.

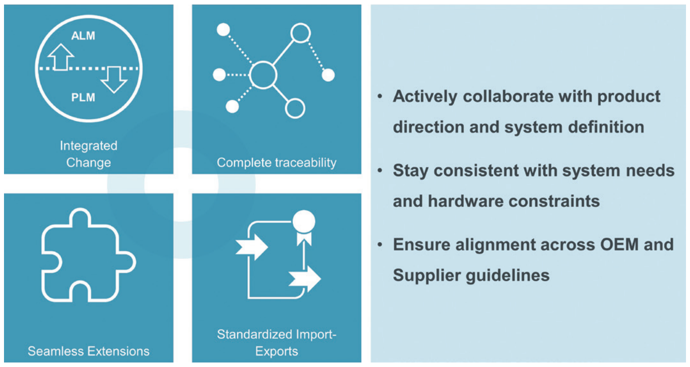
ALM
PLM
Integrated Change
Complete traceability
Seamless Extensions
Standardized Import-Exports
• Actively collaborate with product direction and system definition
• Stay consistent with system needs and hardware constraints
• Ensure alignment across OEM and Supplier guidelines

Figure 6: Advanced application engineering management solutions can consume system requirements to create a direct link between the system and application levels.

This is can be referred to as the product integration stage, as it integrates the overall product and software development cycles. Product integration allows active collaboration between the software definition and planning, and the overall product direction and system definitions. Such collaboration ensures that embedded software applications are consistent with hardware constraints and the high-level system needs. This also helps OEMs and suppliers align their respective guidelines and goals.

Requirements, tests, and targets

The system level definitions and hardware constraints are codified into system level requirements. With these requirements, software engineers can decompose the embedded application level requirements, noise factors, failure modes and effects analyses (FMEAs), test cases, and functional goals. These requirements are specific to each embedded software application, but remain consistent with the system-level context.

To aid with regulatory and process compliance, software engineers can manage application risk using ISO26262 guidelines and Automotive-SPICE. The engineers can also use the Polarion platform to manage safety goals, functional safety concepts, and related safety requirements with full traceability all the way upstream to system context and downstream to lines of code and models. This enables the engineering teams to reduce the number of deviations, non-conformance and recalls.

With a unified embedded application development platform, users can seamlessly collaborate with all key stakeholders, and their deliverables, regardless of the tool used to develop each deliverable (figure 7). With such collaboration, the development of application deliverables, such as models or test plans, will remain consistent despite the continuous changes in software requirements or hardware implementations. The collaboration provided by the application development platform can also capture specifications to help teams to optimize each application for several ECU hardware variants and operating systems.

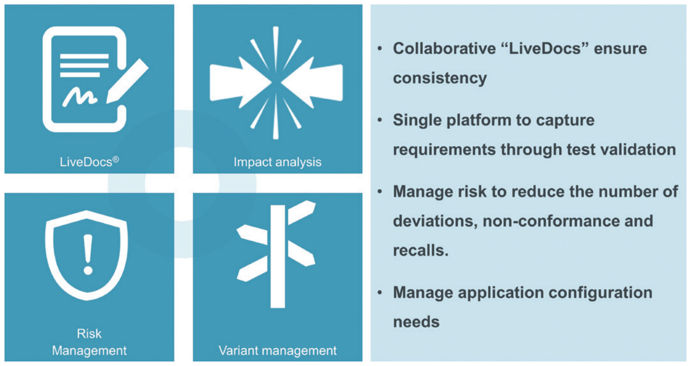
LiveDocs®
Impact analysis
Risk Management
Variant management
• Collaborative "LiveDocs" ensure consistency
• Single platform to capture requirements through test validation
• Manage risk to reduce the number of deviations, non-conformance and recalls.
• Manage application configuration needs

Figure 7: A unified application development platform enables collaboration and consistency among teams developing application models, test plans, and more.

Architecture and modeling

While software architects work on the architecture and modeling stage, they relate to the application-specific requirements, specifications, and behaviors, to optimize the software architecture by creating software models and capturing interactions between software components (figure 8).

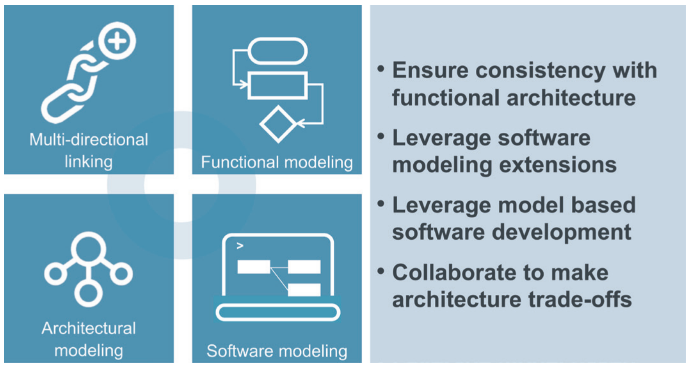
Multi-directional linking
Functional modeling
Architectural modeling
Software modeling
• Ensure consistency with functional architecture
• Leverage software modeling extensions
• Leverage model based software development
• Collaborate to make architecture trade-offs

Figure 8: A unified application engineering platform helps engineers to coordinate architecture and modeling tasks, make trade-off analyses, and leverage model-based software development.

A model-based software development approach best supports software engineers as they work under these pressures. By starting with robust models, engineers can develop a rich architecture that includes potential behavior of the final software. A model-based approach also enables editing, refactoring, enriching, and analysis of the architecture at the pace required to manage these pressures.

Architecture analysis, test, and verification can be moved earlier in the development cycle. As tests are run, findings can be incorporated back into the software models. Interoperability with other simulation solutions, such as a mechatronics simulation, through functional mock-up interfaces (FMIs) or with Simulink creating closed-loop from testing and verification back to the software models.

Polarion orchestrates the creation and modification of these deliverables by connecting directly with industry standard tools such as Embedded Software Designer (ESD), MATLAB/Simulink, Enterprise Architect, and others. As the models and component interactions are developed and become mature, Polarion can link them back to the application definition. This ensures that all the necessary requirements, specifications, and behaviors have a model representation with robust architecture that connects all the necessary software component interactions. This step validates the software model before any code is implemented.

The architecture and modeling stage facilitates the design and validation of vehicle software applications with a common environment for system and software engineers with multi-directional requirements traceability. The result is a digital twin of the software architecture and individual applications that will inform and reflect downstream design decisions. This stage also serves to ensure consistency and collaboration across the functional architecture, application behavior models, and requirements. Various teams can make architectural trade-offs and leverage software models to identify issues, observe variabilities in behavior, and examine semantics.

By unifying the application definition and planning stage on a single platform, software teams can assign tasks for modeling, coding, test-execution, build production, and more across the needed toolsets. At any given time, software engineers can peek into the system level definitions and constraints and collaborate with other system users.

Furthermore, the creation of, and any subsequent changes to, detailed software requirements will trigger software architecture and modeling changes. With the refined requirements, software engineers can update these models to align with control algorithms. The software engineers can also execute model-in-the-loop (MiL) tests to verify and validate that desired outcomes are achieved at the application and system level before any code level changes are triggered.

Conclusion

The software content of modern vehicles is growing at an alarming rate to accommodate the demands of a changing automotive industry. Consumers expect ever-greater functionality from their vehicles through advanced safety features, vehicle connectivity, and customizable experiences. Other more fundamental vehicle systems have been under the domain of software for a few years: power steering, engine management, and braking systems to name a few. These individual functions may be well understood, but the increasing interactions between these and other vehicle systems are introducing new challenges.

Increasing complexity from greater software content, sophistication, and cyber-physical interdependence requires new levels of orchestration for the product and application lifecycle. Such coordination demands a powerful solution, employing a unified product-software digital thread, to trace from the highest-level system definitions down to individual implementations (figure 9).

With such a solution, software engineering teams can perform application development tasks with explicit knowledge of the system context for each deliverable. Teams can collaborate to make architecture trade-offs and manage change with increased agility. Such a solution also enables a closed-loop from verification and validation back into application definition, planning, and development to continuously optimize and improve quality.

The application definition and planning phase, however, is just the first step in developing the sophisticated software applications needed to enable the increasingly computerized vehicles of today. Once planning and definitions are complete, software engineers must implement the models into actual code and test to ensure high quality. The applications also must be configured to match the hardware and operating system variants in the vehicles in which they will be deployed. These following sub-processes stand to benefit equally from a unified embedded application development platform.

Ultimately, companies that are able to effectively manage the development of software applications across organizations, engineering domains, and functional abstractions will be in the best position to thrive in the digitalized automotive industry. This will be especially true as software functionality becomes less dependent on ECU hardware due to an increased use of vehicle operating systems and firmware. A robust methodology to track each application, its functional contents, system constraints, and potential hardware constraints will be key to success. A unified platform for automotive embedded application development is the foundation on which this capability will be built.

To read more about how Siemens solutions can help you, please visit siemens.com/aes where you will find additional content such as blogs, whitepapers, podcasts, product videos, webinars, solution capabilities and infographics.

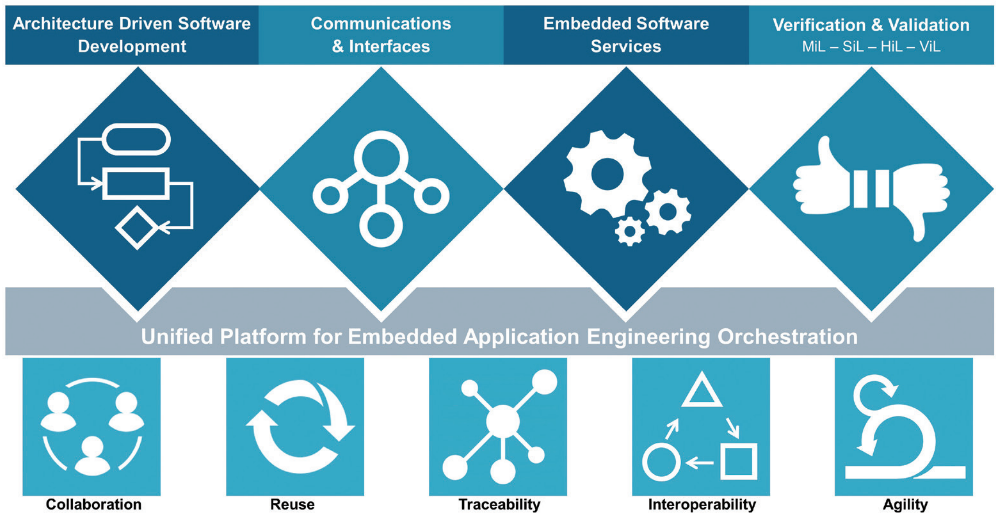
Architecture Driven Software Development
Communications & Interfaces
Embedded Software Services
Verification & Validation
MiL – SiL – HiL – ViL
Unified Platform for Embedded Application Engineering Orchestration
Collaboration
Reuse
Traceability
Interoperability
Agility

Figure 9: A unified platform for embedded application engineering orchestration provides traceability and facilitates collaboration through all processes of application development.

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Our solutions help companies of all sizes create and leverage digital twins that provide organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

siemens.com/software

81120-C4 12/19 C